PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For PolyPup Ball

29 July 2021

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

1 Overview

This report has been prepared for PolyPup Ball. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

At the time of assessment, the contracts were sent to Paladin via a GitHub repository and may differ from the ones deployed on the blockchain.

1.1 Summary

Project Name	PolyPup Ball
URL	https://polypup.finance/
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`BallToken`	`BallToken.sol`	PENDING
`MasterChef`	`MasterChef.sol`	PENDING
`GitHub`	`https://github.com/PolyPup-Farm/contracts-ball/`	

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🔴 High	1	1	-	-
🟠 Medium	2	2	-	-
🟡 Low	3	3	-	-
🟣 Informational	5	5	-	-
Total	**11**	**11**	**0**	**0**

Classification of Issues

Severity	Description
🔴 **High**	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 **Medium**	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 **Low**	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 **Informational**	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 BallToken

ID	Severity	Summary	Status
01	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	✓ RESOLVED
02	INFORMATIONAL	`delegateBySig` can be frontrun and cause denial of service	✓ RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
03	HIGH	Severely excessive rewards issue when a token with a transfer tax is added	✓ RESOLVED
04	MEDIUM	`StartBlock` cannot be adjusted after pools have been added	✓ RESOLVED
05	MEDIUM	Setting `feeAddress` to the zero address will break most functionality	✓ RESOLVED
06	LOW	Adding an EOA or a non-token contract as a pool will break `updatePool` and `massUpdatePools`	✓ RESOLVED
07	LOW	The `pendingBall` function will revert if `totalAllocPoint` is zero	✓ RESOLVED
08	INFORMATIONAL	dev function can be renamed and made `external`	✓ RESOLVED
09	INFORMATIONAL	Pools use the contract balance to figure out the total deposits	✓ RESOLVED
10	INFORMATIONAL	`add`, `set`, `deposit`, `withdraw`, `emergencyWithdraw`, `updateEmissionRate` and `setFeeAddress` can be made `external`	✓ RESOLVED
11	INFORMATIONAL	Lack of events for `add`, `set` and `updateStartBlock`	✓ RESOLVED

2 Findings

2.1 BallToken

2.1.1 Token Overview

The contract allows for BALL tokens to be minted when the `mint` function is called by the Owner, who at the time of deployment would be the deployer. However, ownership is generally transferred to the Masterchef via the `transferOwnership` function for emission rewards to be minted and distributed to users staking in the Masterchef. The `mint` function can be used to pre-mint tokens for various uses including injection of initial liquidity, token presale, airdrops, and others.

Address	TBC
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None

2.1.2 Privileged Roles

The owner of the BallToken contract should be transferred to the Masterchef. The following functions can be called by the owner of the contract:

- `mint`
- `renounceOwnership`
- `transferOwnership`

2.1.3 Issues & Recommendations

Issue #01	**`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef**
Severity	● LOW SEVERITY
Description	The `mint` function could be used to pre-mint tokens for legitimate uses including, but not limited to, the injection of initial liquidity, token presale, or airdrops; however, this function may also be used to pre-mint and dump tokens when the token contract has been deployed but before ownership is set to the Masterchef contract. This risk is prevalent amongst less-reputable projects, and any pre-mints can be prominently seen on the Blockchain.
Recommendation(s)	Consider being forthright if this `mint` function is to be used by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of the mints.
Resolution	✓ RESOLVED The client has stated that 4000 tokens will be pre-minted to provide the initial liquidity, after which the ownership of the token will be transferred to the Masterchef. We will update this report when we confirm this has been done.

Issue #02	`delegateBySig` **can be frontrun and cause denial of service**
Severity	INFORMATIONAL
Description	Currently if `delegateBySig` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up `delegateBySig` transactions in the mempool and execute them before a contract can. The issue with this is that the rest of said contract functionality would be lost as well. This could be a problem in case it would have been executed by a contract that would have rewarded you for your delegation for example.
Recommendation(s)	Consider adding the desired message sender in the `structhash` and requiring this desired sender to be equal to `msg.sender`. This reduces the problem to having the message sender be able to frontrun you, which is okay if it is a reviewed contract.
Resolution	✓ RESOLVED The client has said they will carefully consider this behavior if they ever implement `delegateBySig` in derivative contracts.

2.2 MasterChef

The Masterchef is a fork of Goose Finance's Masterchef. A notable feature of forking this Masterchef is the removal of the migrator function from Pancakeswap, which of late has been used maliciously to steal users' tokens. We commend Polypup Ball on their decision to fork a relatively safer version of the Masterchef.

2.2.1 Privileged Roles

The following functions can be called by the owner of the Masterchef:

- `add`
- `set`
- `dev`
- `setFeeAddress`
- `updateEmissionRate`
- `updateStartBlock`
- `renounceOwnership`
- `transferOwnership`

2.2.2 Issues & Recommendations

Issue #03	Severely excessive rewards issue when a token with a transfer tax is added
Severity	● HIGH SEVERITY
Description	When tokens with a transfer tax are added to the pools, this will result in significant excessive rewards. Due to the way the Masterchef handles rewards, rewards can be heavily inflated when the balance of the Masterchef no longer matches that of user deposits. This happens for example with transfer tax tokens. This flaw of the Masterchef has recently been exploited on a significant number of projects, all of which their native tokens went to $0 afterwards. This issue was also present in SushiSwap (the original Masterchef). Since they were never meant to have any tokens but LP tokens, it was not a problem there but has become a problem to projects who have started forking it for usage with less standard tokens.

Recommendation(s)	Consider using the current standard of handling deposits, which is based on how Uniswap handles transfer fees:
	`uint256 balanceBefore = pool.lpToken.balanceOf(address(this));`
	`pool.lpToken.transferFrom(msg.sender, address(this), _amount);`
	`_amount = pool.lpToken.balanceOf(address(this)).sub(balanceBefore);`
	
	Note that by using this method, you can also add the specific transfer tax logic for the native token if you so wish.

Resolution	✓ RESOLVED

Issue #04	StartBlock cannot be adjusted after pools have been added
Severity	● MEDIUM SEVERITY
Description	The updateStartBlock function can only be called if no pools have been added. If there are any unforeseen circumstances that require a delay to the project, then the Masterchef contract would have to be redeployed if pools have already been added. Consider giving your project the flexibility to update StartBlock, with a for loop to update lastRewardBlock in poolInfo as well.
Recommendation(s)	Consider implementing the following for updateStartBlock to give your project the flexibility to adjust StartBlock :

```
function updateStartBlock(uint256 _newStartBlock) external onlyOwner {

    require(block.number < startBlock, "cannot change start block if
farm has already started");
    require(block.number < _newStartBlock, "cannot set start block in
the past”);
    uint256 length = poolInfo.length;

    for (uint256 pid = 0; pid < length; ++pid) {
        PoolInfo storage pool = poolInfo[pid];
        pool.lastRewardBlock = _newStartBlock;
    }
    startBlock = _newStartBlock;
}
```

Note that this loop may run out of gas if a very significant amount of pools are added.

Resolution	✓ RESOLVED

Issue #05	Setting `feeAddress` to the zero address will break most functionality
Severity	● MEDIUM SEVERITY
Description	Within the token contract, minting or transferring tokens to the zero address will revert the transaction. Deposits and withdrawals will break if the `feeAddress` is ever set to the zero address. Harvesting will fail as well.
Recommendation(s)	To prevent this from ever happening by accident and to limit governance risks, consider adding a requirement like `require(_feeAddress != address(0), “!nonzero”);` to the configuration function.
Resolution	✓ RESOLVED

Issue #06	**Adding an EOA or a non-token contract as a pool will break `updatePool` and `massUpdatePools`**
Severity	● LOW SEVERITY
Description	`updatePool` will always call `balanceOf(address(this))` on the token of this pool.
Recommendation(s)	Consider simply adding a test line in the add function. If the token does not exist, this will make sure the add function fails. `_lpToken.balanceOf(address(this));`
Resolution	✓ RESOLVED

Issue #07	**The `pendingBall` function will revert if `totalAllocPoint` is zero**
Severity	● LOW SEVERITY
Description	In the `pendingBall` function, at some point a division is made by the `totalAllocPoint` variable. If all pools have their rewards set to zero, this variable will be zero as well. The requests will then revert with a division by zero error.
Recommendation(s)	Consider only calculating the accumulated rewards since the `lastRewardBlock` if the `totalAllocPoint` variable is greater than zero. This check can simply be added to the existing check that verifies the `block.number` and `lpSupply`, like so: `if (block.number > pool.lastRewardBlock && lpSupply != 0 && totalAllocPoint > 0) { … }`
Resolution	✓ RESOLVED

Issue #08	dev function can be renamed and made `external`
Severity	● INFORMATIONAL
Description	Users may be confused on what the `dev` function does, as its associated event is currently declared as `setDevAddress`.
Recommendation(s)	Consider renaming the function to `setDevAddress`, and make it `external`.
Resolution	✓ RESOLVED

Issue #09	Pools use the contract balance to figure out the total deposits
Severity	● INFORMATIONAL
Description	As with pretty much all Masterchefs, the total number of tokens in the Masterchef contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the Masterchef. More severely, because the native token is constantly minted, this will cause severe dilution on the native token pool.
Recommendation(s)	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits.
Resolution	✓ RESOLVED

Issue #10	add, set, deposit, withdraw, emergencyWithdraw, updateEmissionRate and setFeeAddress can be made external
Severity	INFORMATIONAL
Description	add, set, deposit, withdraw, emergencyWithdraw, updateEmissionRate and setFeeAddress functions can be changed from public to external. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases (https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices).
Recommendation(s)	Consider making these functions external.
Resolution	✓ RESOLVED

Issue #11	Lack of events for add, set and updateStartBlock
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation(s)	Add events for add, set and updateStartBlock.
Resolution	✓ RESOLVED

PALADIN
BLOCKCHAIN SECURITY